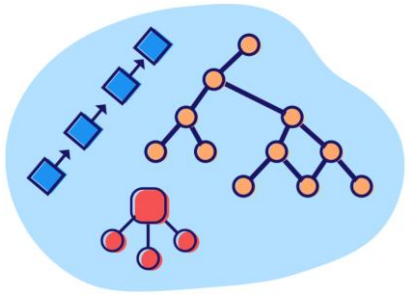
الجامعة العربية الخاصة للعلوم والتكنولوجيا
كلية الهندسة المعلوماتية
قسم تقانة المعلومات

Algorithms & Data Structures (2)

Lecture-4: Hashing Tables

(قوائم التقطيع)



Dr. Asmaa Shaar

فهرس المحاضرة

- مقدمة
- مفاهيم أساسية
- خطوات حشر عنصر في Hash Table
- طرق التقطيع – Hashing Methods
- طرق حل التصادمات (Collision Resolution Methods)
- أمثلة محلولة

مقدمة

- تتطلب تقنيات البحث التقليدية **عدة مقارنات** قبل أن نتمكن من العثور على العنصر المطلوب.
- **باستخدام المصفوفات (Arrays):**
  - البحث التسلسلي O(n)
  - البحث الثنائي O(log n)
- **باستخدام شجرة البحث الثنائية (Binary search tree)**
  - البحث O(log n)
- **باستخدام جدول التقطيع (Hashing Table):**
  - يتم العثور على البيانات **باختبار واحد فقط** ← نعرف بالضبط مكان تواجد البيانات مسبقاً ونذهب مباشرة إلى هناك ← O(1) بحث مثالي.

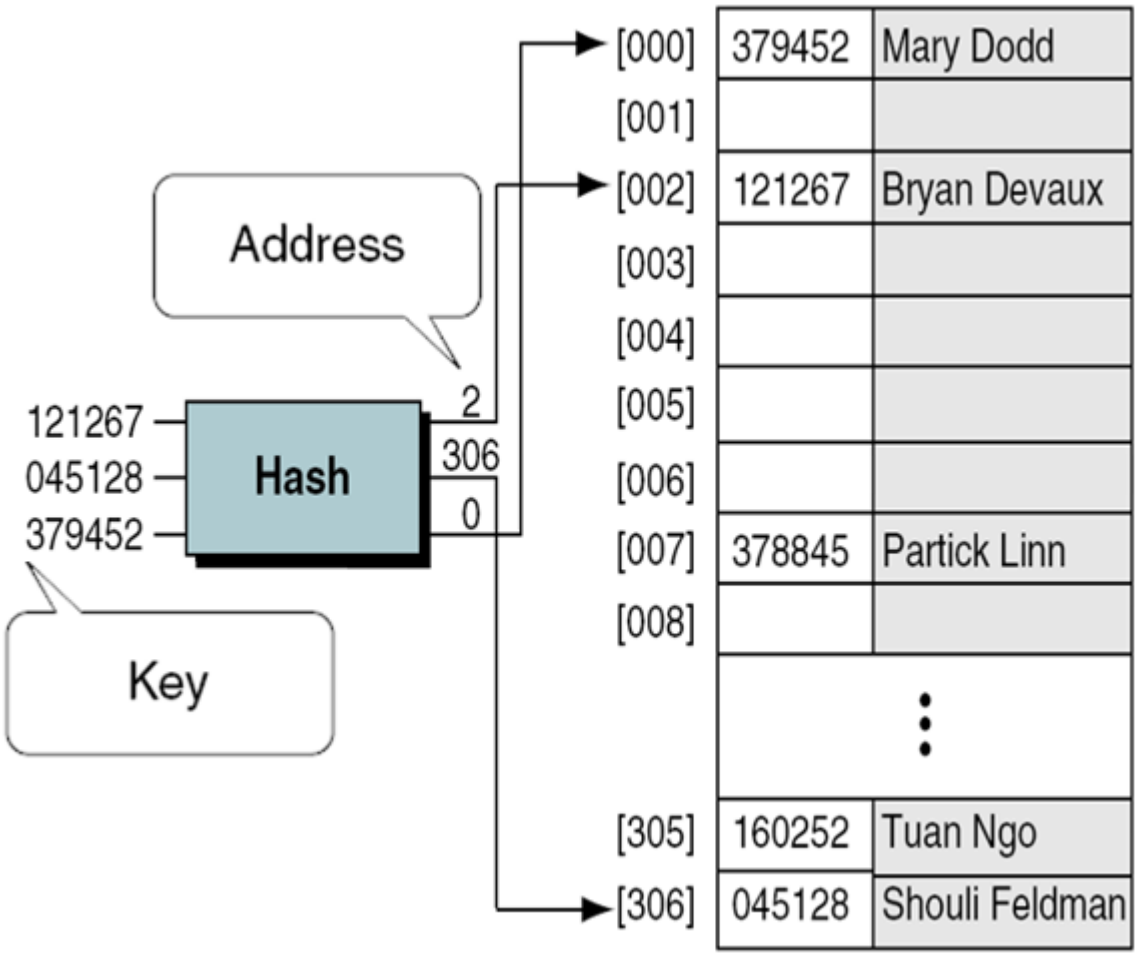
مفاهيم أساسية

- لتحقيق Hashing Tables يمكن استخدام **المصفوفات**. بحيث يمكن أن يكون **نمط معطيات** العناصر المخزنة:
  - **بسيطاً** مثل عدد أو محرف أو سلسلة الخ...
    - ✓ نعتبر هذه القيمة التي نبحث عنها هي **مفتاح البحث (key)**.
  - **مركّب** مثل السجلات
    - ✓ لإيجاد سجل معيّن يتم البحث عن إحدى الحقول وهو يكون **مفتاح البحث (key)**.
- يتم تخزين المعطيات في المصفوفة من خلال تطبيق علاقة (تابع تقطيع) على **مفتاح البحث (key)**

  - يقوم تابع تقطيع بتحويل المفتاح إلى **قيمة عددية** تمثل **عنوان** العنصر (الدليل في المصفوفة)
  - يتم استخدام نفس تابع التقطيع السابق عند البحث في **جدول التقطيع**.

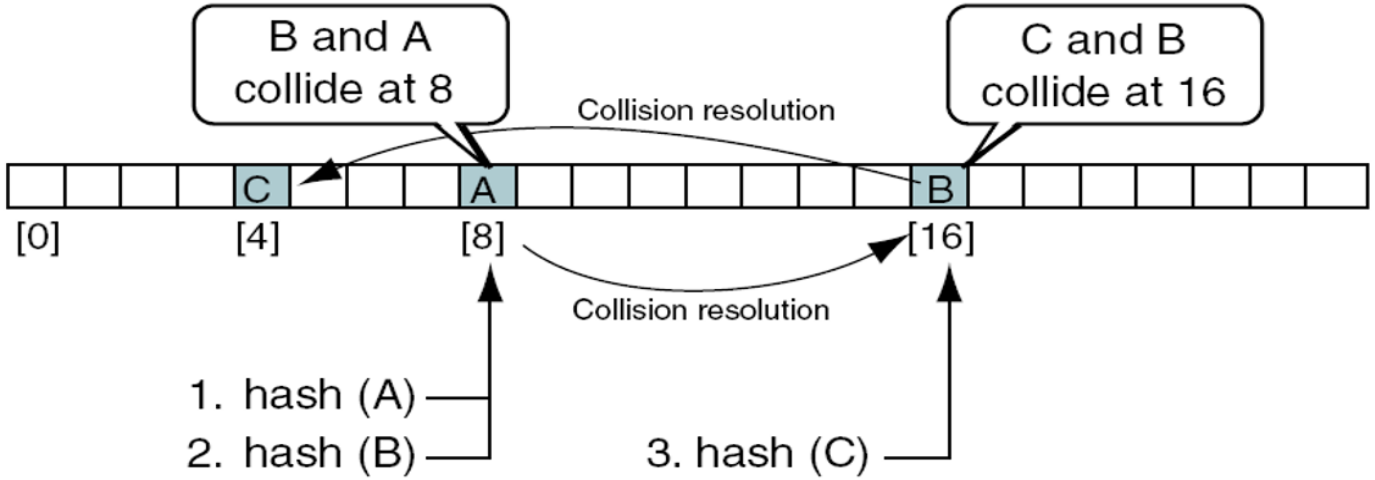
مفاهيم أساسية

- مثال: بفرض لدينا تطبيق يدير سجلات للطلاب (رقم الطالب، اسم الطالب) .
  - عندما نريد البحث عن بيانات طالب محدد باستخدام رقم الطالب (كمفتاح بحث):
    - ✓ أولاً يجب أن تتم عملية حشر سجلات الطلاب في المصفوفة باستخدام تابع Hashing ما.
    - ✓ وعند البحث يجب استخدام نفس تابع الـ Hashing المستخدم وقت الحشر.



مفاهيم أساسية

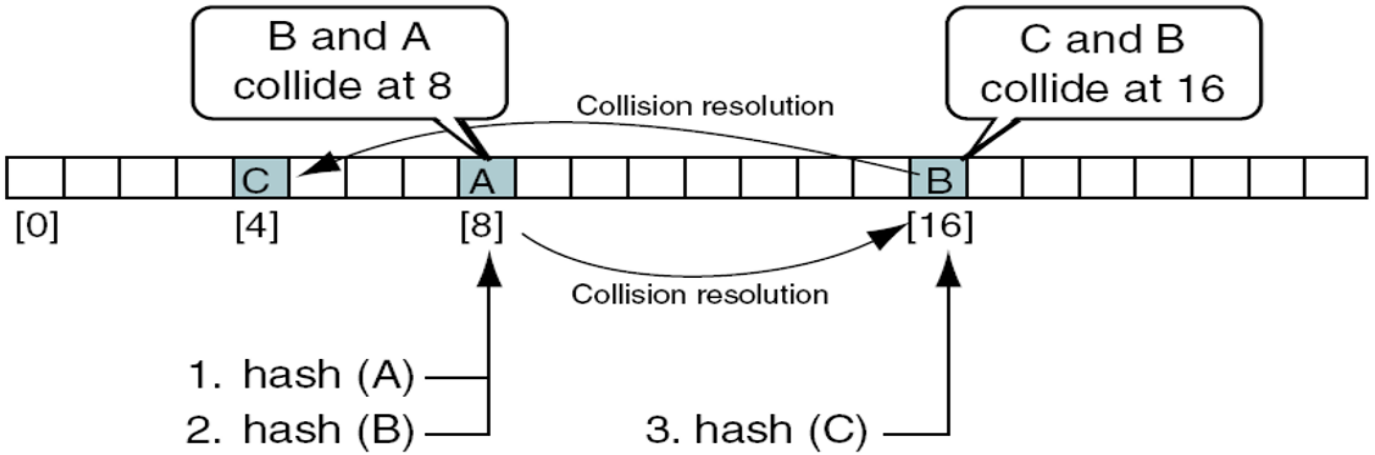
- اذا تم تقطيع مفتاح وحصلنا بالصدفة على **عنوان محجوز مسبقاً** من قِبل مفتاح آخر يحصل **تصادم** على ذلك العنوان (collision)، بحيث:
  - ← يجب علينا حساب **عنوان بديل للمفتاح الجديد** عن طريق تطبيق إحدى طرق حل التصادمات (collision resolution method).



- **ملاحظة**: لحل التصادم بطريقة **أسرع**، يفضل أن يكون لدينا **خانات إضافية** في المصفوفة أكثر من **عدد العناصر** المطلوب تخزينه.
  - عادة يُنصح بحجز طول المصفوفة بحيث نستهلك منها فعلياً 75% من العناصر فقط أما بقية العناصر (25%) سوف تبقى فارغة.

مفاهيم أساسية

- **تعاريف:**
  - home address: **أول عنوان** ناتج عن تقطيع مفتاح
    - ✓ بحيث يلزم أحياناً إيجاد عدة **عناوين بديلة** بسبب التصادمات.
  - probe: فحص العنوان الناتج عن **تقطيع المفتاح** لمعرفة إن كان **فارغاً** أم لا، أو إن كان محتواه مطابقاً مع المفتاح أم لا.
  - collision: تصادم = هناك **عنوان مشترك** لمفتاحين مختلفين أو أكثر
    - ✓ يمكن أن يحصل **تصادم على العناوين البديلة** أيضاً.
  - synonyms: مرادفات = **مفاتيح مختلفة** لها نفس home address، أي اصطدمت مع بعض.
  - double hashing: تطبيق أكثر من **تابع تقطيع واحد** على نفس المفتاح بهدف **تخفيف نسبة التصادمات**.



خطوات حشر عنصر في Hash Table

- خطوات حشر عنصر في Hash Table **دون تكرار قيمة المفتاح**.

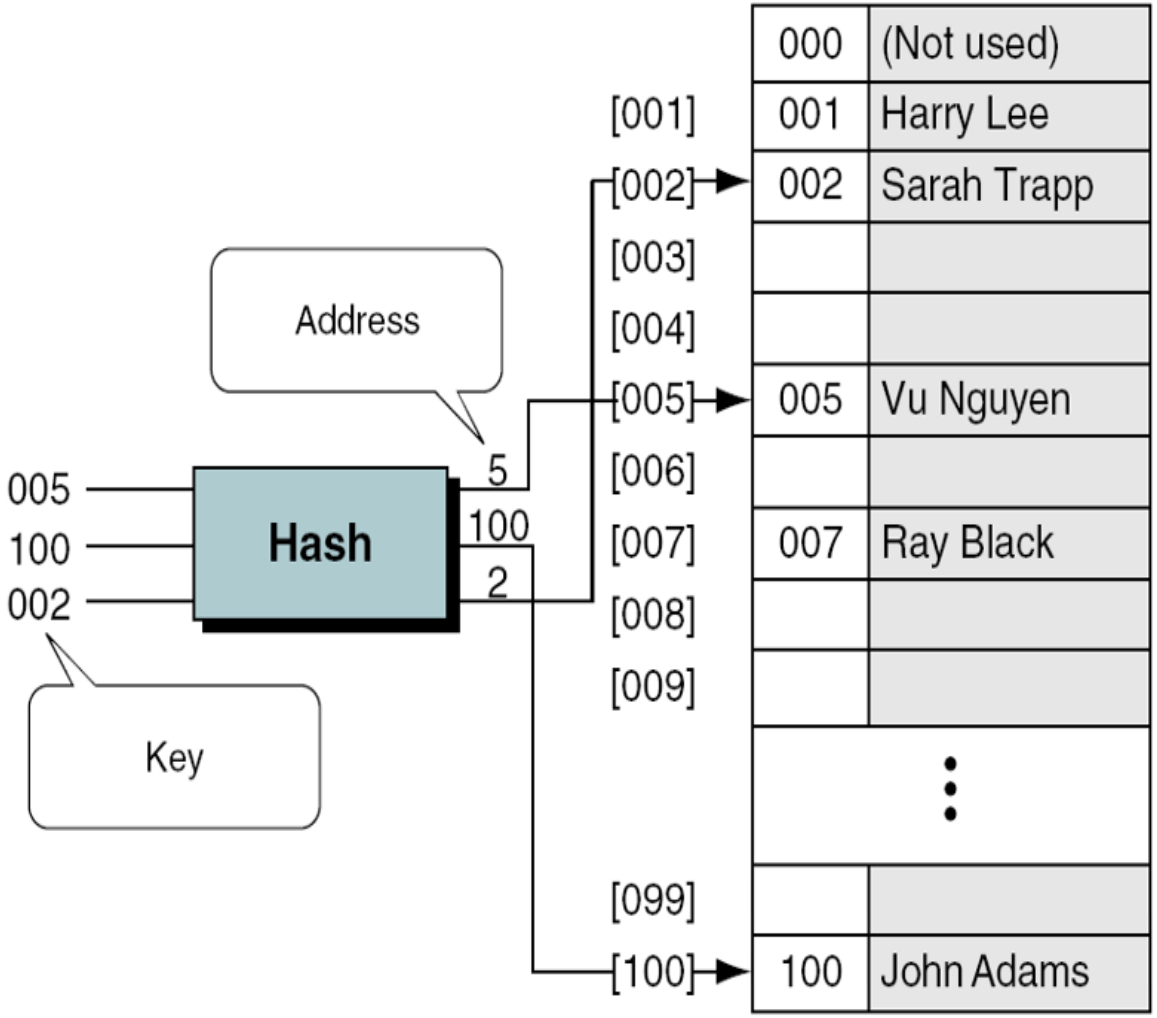
1. يتم تقطيع المفتاح بإحدى طرق التقطيع (Hashing Methods).
2. يتم فحص العنوان الناتج x لمعرفة إن كان مشغولاً أم لا.

- إذا كان العنوان x غير مشغول ← نتابع هكذا إلى أن نوجد العنصر أو نتأكد تماماً من عدم وجوده وعندها نحشره فعلياً في العنوان x.
- فإذا كان العنوان x مشغولاً (= تصادم) نقارن المفتاحين المرادفين.
  - إذا كانا مفتاحين **متطابقين** ← **المفتاح موجود** ونوقف عملية الحشر الجديدة.
  - أما إذا كانا مفتاحين **مختلفين** ← نتابع هكذا إلى أن نوجد العنصر أو نتأكد تماماً من عدم وجوده وعندها نحشره فعلياً في **أول عنوان غير مشغول** حصلنا عليه أثناء الاختبارات السابقة.

Hashing Methods – طرق التقطيع

طرق ربط العنصر مع الدليل في المصفوفة

1- الطريقة المباشرة (Direct Method)



- يكون المفتاح هو **العنوان نفسه دون أي معالجة حسابية**.
- يجب أن تحتوي المصفوفة على خانة مقابل كل مفتاح.
- طريقة **قوية للغاية** لأنها تضمن عدم حصول تصادمات عندما تكون المفاتيح مختلفة عن بعض.
- هذه الطريقة هي **الأبسط** ولكن لا يفضل استخدامها إلا في **حالات محدودة جداً** (إذا كان حجم المعطيات محدود وكانت قيم المفاتيح شبه متتالية ومن دون تكرار)
  - مثل أيام الشهر أو أرقام محدودة لطلاب أو موظفين الخ..

2- طريقة الطرح (Subtraction Method)

- تكون **المفاتيح متتالية** ولكنها لا تبدأ من 1.
- في هذه الحالة **نطرح قيمة ثابتة** (const) من المفاتيح لتصغير القيم والعدد الناتج هو العنوان المطلوب.

address = key – const

- على سبيل المثال، قد يكون لدى شركة 100 موظف فقط، ولكن تبدأ أرقام الموظفين من 1001 و تصل إلى 1100. ← في هذه الحالة يقوم تابع التقطيع بطرح 1000 من المفتاح لتحديد العنوان.
  - أمثلة:

hash(1032)= 1032 - 1000 = 32
hash(1100)= 1100 - 1000 = 100

- طريقة بسيطة جداً وتضمن **عدم حدوث تصادمات** عندما تكون المفاتيح مختلفة عن بعض.
- ولكن كما ذكرنا في الطريقة المباشرة يفضل استخدامها في **حالات محدودة جداً**

3- طريقة باقي القسمة (Modula Division Method)

- العنوان المطلوب يساوي باقي قسمة المفتاح على طول المصفوفة:

address = key mod list_size

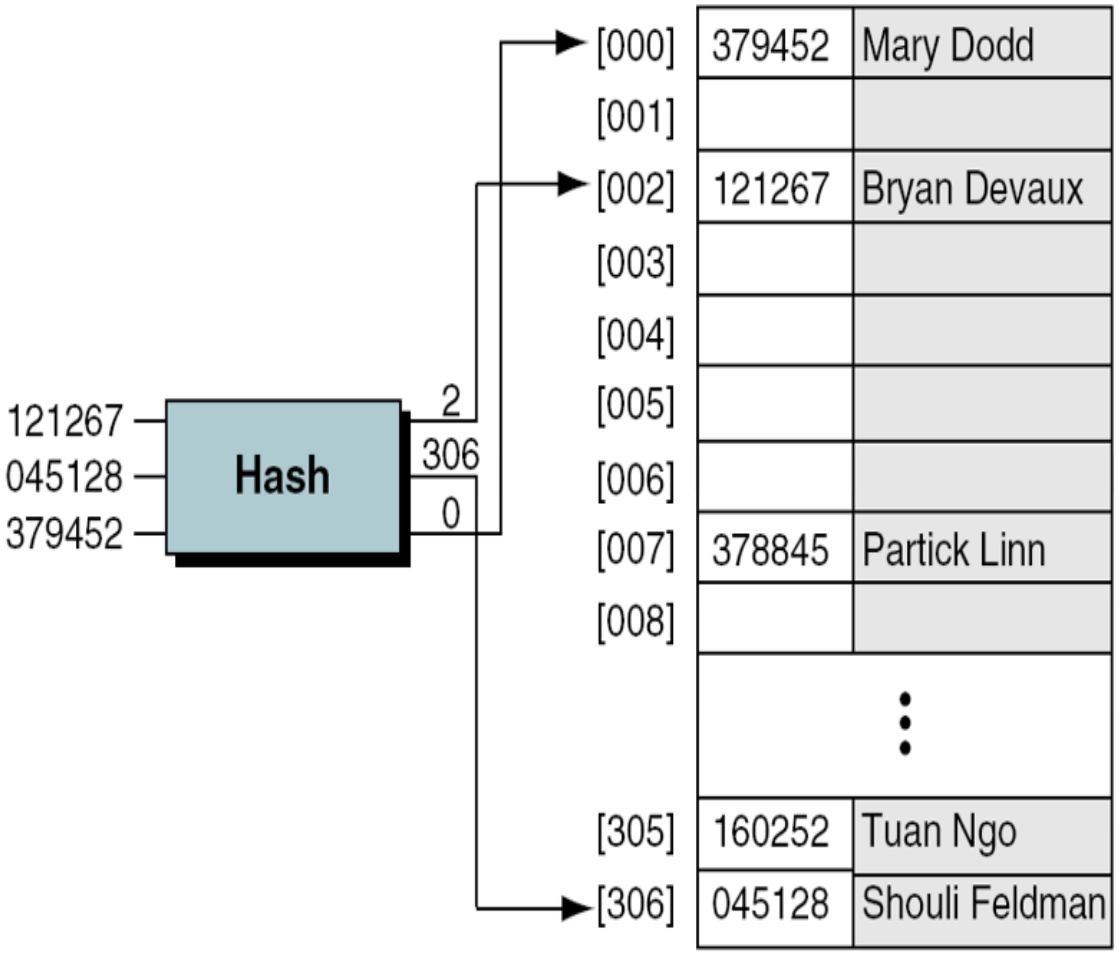
- في هذه الطريقة ممكن أن تحصل **تصادمات** حتى ولو كانت المفاتيح **مختلفة تماماً**.

- تعمل هذه الطريقة مع أي حجم مصفوفة، ولكن عندما يكون حجم المصفوفة **رقماً أولياً** (prime number) ينتج **تصادمات أقل**.
  - لذلك يُنصح، إن أمكن حجز طول المصفوفة رقماً أولياً.

- مثال: لو كان لدى شركة 225 موظف ونريد الآن تحديد أنسب طول للمصفوفة:
  - نرجع أولاً إلى فكرة الـ 75% من العناصر للاستهلاك و 25% عناصر فارغة (من أجل سهولة معالجة التصادمات)، فيكون مثلاً 300 هو طول المصفوفة المناسب.
  - ولكن **أول رقم أولي أكبر** من 300 هو 307 ← لذلك يُفضل أن نختار 307 كحجم المصفوفة (العناوين تتراوح من 0 إلى 306).

3- طريقة باقي القسمة (Modula Division Method)

- مثال: مصفوفة الموظفين الجديدة وبعض عناوينها المقطعة في الشكل التالي:



address = hash(121267) = 121267 mod 307 = 2
address = hash(045128) = 045128 mod 307 = 306
address = hash(157186) = 157186 mod 307 = 2 → collision

4- طريقة استخراج الأرقام (Digit Extraction Method)

- يفضل استخدام هذه الطريقة في حال **مفاتيح طويلة جداً** مثل الرقم الوطني أو أرقام المحمول الخ.
- يتم استخراج **خانات محددة** من المفتاح واستخدامها كعنوان (نفس الخانات من جميع المفاتيح).
  - مثال: لو كان لدينا مفتاح مؤلف من 6 أرقام، يمكننا تحديد الخانة الأولى والثالثة والرابعة (من اليسار) لنحصل على عنوان ثلاثي:

address = hash(379452) = 394
address = hash(121267) = 112
address = hash(378845) = 388
address = hash(045128) = 051
address = hash(161204) = 112 → collision

- يمكن حصول تصادمات حتى ولو كانت المفاتيح **مختلفة تماماً**.

4- طريقة استخراج الأرقام (Digit Extraction Method)

- عدد الخانات المستخرجة يجب ألا يتجاوز عدد خانات **عنوان العنصر الأخير** في المصفوفة.
  - مثال: لنفترض عدد المفاتيح الكلي يساوي 300 مفتاح
  - ✓ لو تم حجز 300 عنصر فقط نلاحظ أن **700 عنوان ثلاثي** محتمل جميعها خارج المصفوفة وسيتم ضياع وقت كبير لحساب عناوين بديلة واقعة داخل مجال المصفوفة.
  - ✓ لذلك يفضل حجز مصفوفة طولها 1000 لكي نستفيد من جميع العناوين الناتجة مع أصغر قيمة محتملة 000 وأكبر قيمة محتملة 999
- **القاعدة:** من أجل تخفيف نسبة التصادمات، أولاً يجب معرفة عدد المفاتيح الكلي.
  - عدد المفاتيح بالعشرات: نحجز مصفوفة بطول 100 ووقت التقطيع نستخرج 2 خانة للحصول على عنوان ثنائي.
  - عدد المفاتيح بالمئات: نحجز مصفوفة بطول 1000 ووقت التقطيع نستخرج 3 خانات للحصول على عنوان ثلاثي.
  - عدد المفاتيح بالآلاف: نحجز مصفوفة بطول 10000 ووقت التقطيع نستخرج 4 خانات للحصول على عنوان رباعي.
  - وهكذا ...

5- طريقة الطوي (Fold-Shift)

- نقسم المفتاح إلى أقسام ذات طول **يتوافق** مع طول العنوان المراد الحصول عليه (إما ثنائي أو ثلاثي أو رباعي الخ)، ثم نجمع الأقسام والناتج هو العنوان المطلوب.
  - إذا زاد طول الناتج خانة واحدة **نتجاهل** أول خانة من اليسار
- **مثال:** نريد تقطيع رقم وطني مؤلف من تسعة خانات والحصول على عنوان ثلاثي.
  - نقسم الرقم الوطني إلى ثلاثة أقسام مكونة من ثلاثة أرقام، ثم نجمع هذه الأقسام. إذا كان المجموع الناتج أكبر من 999 ، فإننا نتجاهل الرقم الأول من اليسار.

5- طريقة الطوي (Fold-Shift)

- يمكن حصول تصادمات في هذه الطريقة حتى ولو كانت المفاتيح مختلفة تماماً.
  - مثال:

123456789 → 123 + 456 + 789 = 1368 → address = 368
212032124 → 212 + 032 + 124 = 368 → collision

- من أجل تخفيف نسبة التصادمات يفضل تحديد **طول المصفوفة بشكل أمثلي** كما شرحنا في طريقة استخراج الأرقام , أي عندما يكون:
  - عدد المفاتيح بالعشرات: نحجز مصفوفة بطول 100 ويكون طول كل قسم 2 خانة للحصول على عنوان ثنائي.
  - عدد المفاتيح بالمئات: نحجز مصفوفة بطول 1000 ويكون طول كل قسم 3 خانات للحصول على عنوان ثلاثي.
  - عدد المفاتيح بالآلاف: نحجز مصفوفة بطول 10000 ويكون طول كل قسم 4 خانات للحصول على عنوان رباعي.
  - وهكذا ...

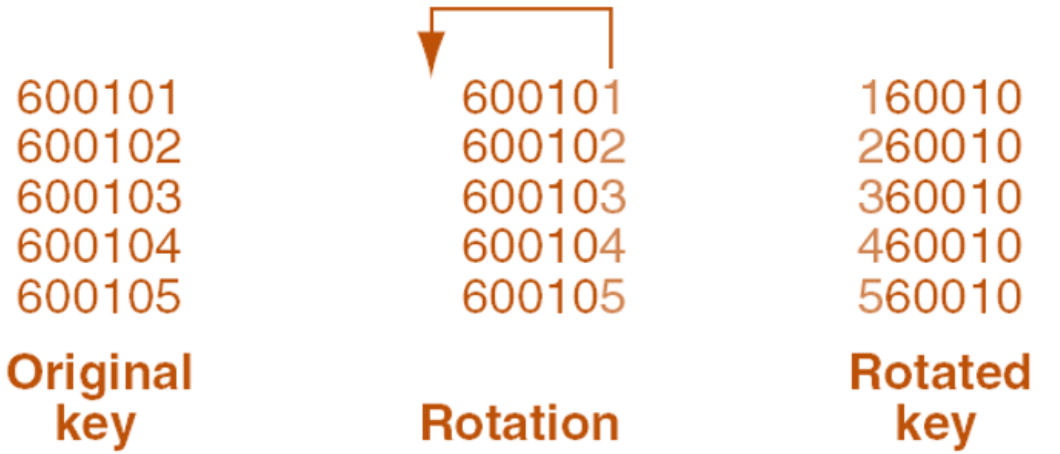
6- طريقة الدوران (Rotation Method)

- بشكل عام، لا يتم استخدام طريقة الدوران بمفردها **بل يتم دمجها** مع طريقة تقطيع أخرى فنحصل على double hashing.
- وعادة تستخدم هذه الطريقة عندما يكون لدينا مفاتيح شبه متطابقة **باستثناء الخانة الأخيرة**، وبالتالي لو طبقنا مثلاً طريقة الطوي Fold-Shift تكون العناوين الناتجة على الأغلب **متسلسلة**، وبالتالي لو حصل تصادم يمكن أن يزداد عدد الـ probes اللازمة للمعالجة.
  - لذلك نستخدم في هذه الحالة طريقة الدوران قبل تطبيق **طريقة التقطيع الأساسية لتفريق عناوين المفاتيح** عن بعضها وهكذا يمكن تخفيف نسبة التصادمات.

Fold-Shift only:
hash(600101) = 600 + 101 = 701
hash(600102) = 600 + 102 = 702
hash(600103) = 600 + 103 = 703
hash(600104) = 600 + 104 = 704
hash(600105) = 600 + 105 = 705

6- طريقة الدوران (Rotation Method)

- يتم الدوران كالتالي: ندوّر الخانة الأخيرة من المفتاح إلى البداية ثم نطبق طريقة تقطيع أخرى على الناتج.
  - مثال:



Rotation + Fold-Shift
hash(160010) = 160 + 010 = 170
hash(260010) = 260 + 010 = 270
hash(360010) = 360 + 010 = 370
hash(460010) = 460 + 010 = 470
hash(560010) = 560 + 010 = 570

Fold-Shift only:
hash(600101) = 600 + 101 = 701
hash(600102) = 600 + 102 = 702
hash(600103) = 600 + 103 = 703
hash(600104) = 600 + 104 = 704
hash(600105) = 600 + 105 = 705

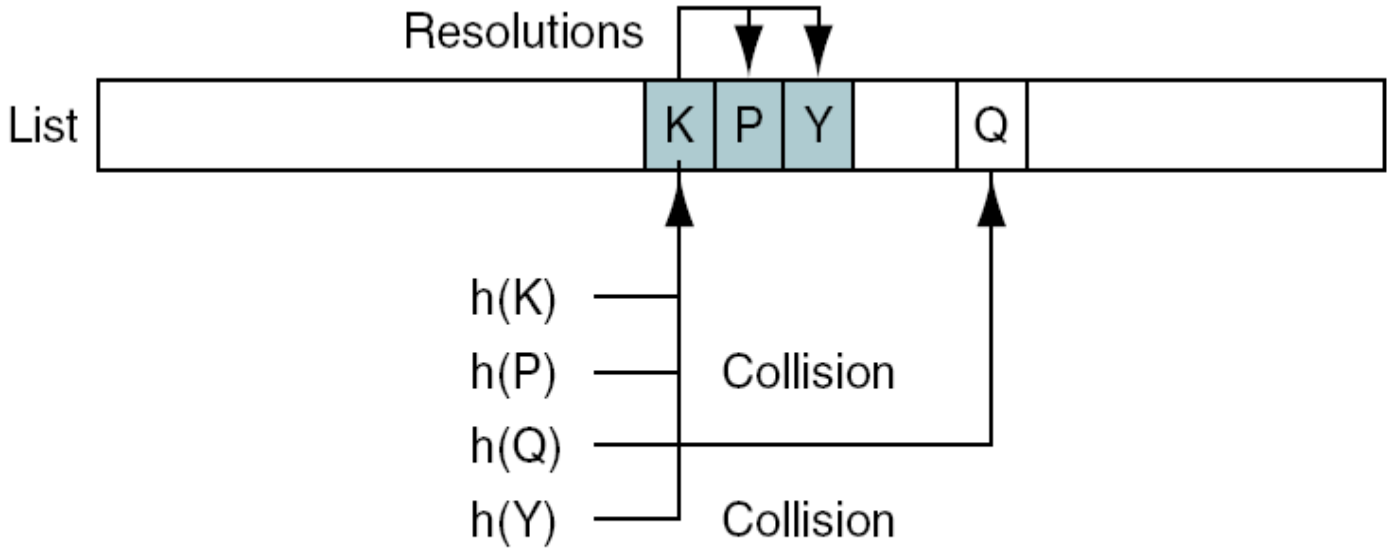
Hashing Methods – طرق التقطيع

- ملاحظة: لو كان المفتاح الذي نريد تقطيعه من **نمط معطيات محرفي (String)**
  - يمكن تحويله إلى قيمة عددية من خلال الاعتماد على ترتيب المحارف في جدول ASCII (A=65 , B=66 , الخ) ومن ثم يمكن تطبيق طريقة التقطيع.
  - فمثلاً لو كان لدينا المفتاح "ABCD" يتحول إلى القيمة العددية "656667768" وهكذا أصبح مناسباً للتقطيع.

طرق حل التصادمات (Collision Resolution Methods)

تعاريف

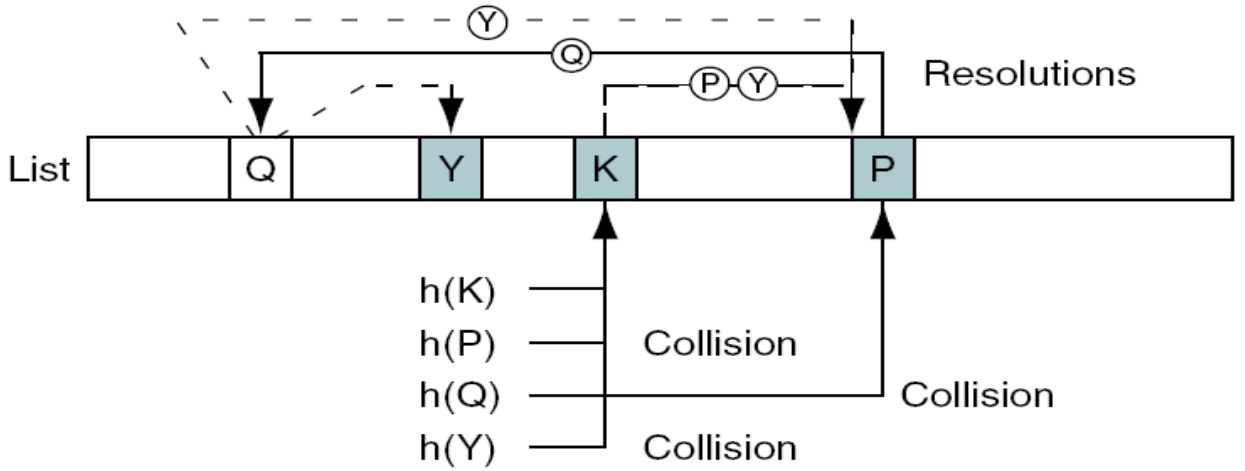
- قبل أن نتعرف على أشهر طرق حل التصادمات يجب أولاً فهم بعض المصطلحات:
  - **collision path**: مسار التصادمات **لنفس المفتاح**.
  - **primary clustering**: يتم تجمّع المرادفات **بالقرب من** الـ home address الخاص بها.
    - ✓ هذا النوع من التجمع **ليس جيداً** في حال حصول **تصادمات كثيرة** لأننا سنحتاج إلى **معالجة أطول** للحصول على عناوين بديلة.



(a) Primary clustering

تعاريف

- قبل أن نتعرف على أشهر طرق حل التصادمات يجب أولاً فهم بعض المصطلحات:
  - **secondary clustering:** يتم تجمّع المرادفات **بشكل متفرق بعيداً عن** الـ home address (أي أن لكل مرادف مسار تصادمات مختلف).
    - ✓ هذا النوع من التجمع **جيد جداً** لأنه يخفف من نسبة حصول التصادمات.



(b) Secondary clustering

  - معامل التحميل (**load factor**): هو **نسبة** امتلاء Hash Table ويُحسب كالتالي: (عدد المفاتيح المخزنة في الجدول تقسيم طول المصفوفة) ضرب 100، أي: $\alpha = (k / listsize) \times 100$
    - ✓ معظم Hash Tables تربط الأداء الجيد ببقاء عامل التحميل في مجال معين (أقل من 75%).

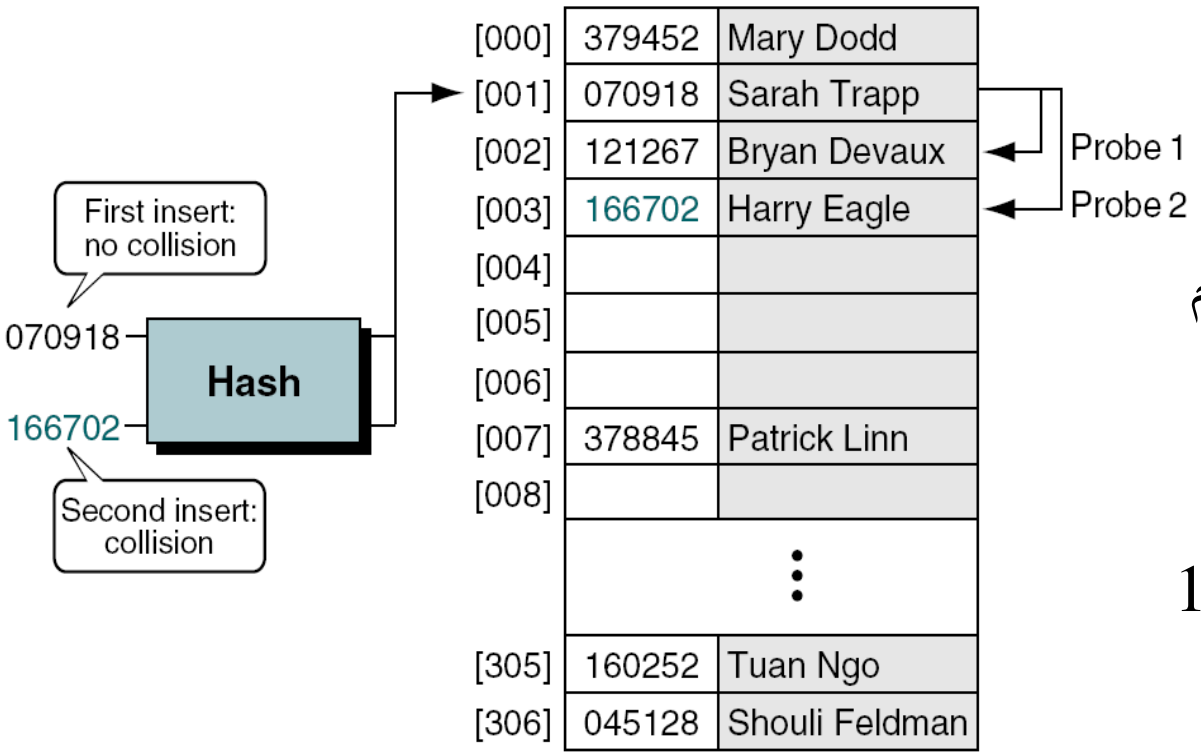
طرق حل التصادمات

- يتحقق ذلك بإحدى الطرق التالية:
  - **باستخدام العنونة المفتوحة (Open Addressing):**
    - ✓ تتضمن كل خانة في جدول التقطيع مفتاح واحد فقط.
    - ✓ اذا حصل تصادم يتم فحص خانات اخرى في جدول التقطيع حتى يتم الوصول لخانة فارغة.
      1. الفحص الخطي (Linear Probe).
      2. الفحص التربيعي (Quadratic Probe).
      3. الفحص بإزاحة للمفتاح (Key Offset Probe) .
  - **باستخدام الرزم (Bucket Hashing).**
    - ✓ تتضمن كل خانة في جدول التقطيع على مؤشر **لقائمة مترابطة من المفاتيح** التي تم تقطيعها في نفس الخانة.

1- الفحص الخطي (Linear Probe)

- نضيف دائماً 1 إلى العنوان الذي تم فيه الاصطدام ونفحص ثانيةً إلى أن نصل إلى أقرب عنوان مفتوح.

New address = old address + 1



- نتوقف عندما نجد عنواناً مفتوحاً أو نستنفد جميع الخانات المحتملة.
  - اذا كانت المصفوفة ممتلئة، عندها يتم رمي استثناء أو توسيع حجم المصفوفة (resize).
- مثال: عند محاولة إدخال المفتاح 166702 باستخدام **طريقة باقي القسمة** يحصل تصادم في العنوان 001 . نحاول حل التصادم عن طريق إضافة 1 إلى العنوان وإدخال البيانات الجديدة في العنوان 002 ، فيحصل تصادم آخر في العنوان الجديد. لذا نضيف 1 مرة أخرى إلى العنوان الأخير وهذه المرة نعثر على عنوان مفتوح 003 ، حيث يمكننا وضع البيانات الجديدة فيه.

2- الفحص التربيعي (Quadratic Probe)

- عند حصول تصادم نضيف للعنوان الحالي تربيع رقم الـ probe الأخير.

New address = old address + probe^2

- بعد التصادم الأول نضيف 1^2 ، أما بعد التصادم الثاني نضيف 2^2 ، وبعد التصادم الثالث نضيف 3^2 ، وهكذا إلى أن نجد عنواناً مفتوحاً أو نستنفد جميع الخانات المحتملة.
- **ملاحظة:** عند خروجنا من نهاية المصفوفة (في كل طرق حل التصادم)، نطبق **باقي القسمة على الناتج** للحصول على عنوان جديد. على سبيل المثال من أجل مصفوفة بطول يساوي 100:

New address = 65 + 1^2 = 66 → collision
New address = 66 + 2^2 = 70 → collision
New address = 70 + 3^2 = 79 → collision
New address = 79 + 4^2 = 95 → collision
New address = 95 + 5^2 = 20 → collision
New address = 20 + 6^2 = 56 → collision etc…

- من الإيجابيات أن نوع التجمع مع استخدام الفحص التربيعي يكون **secondary clustering**
- المشكلة هنا أننا ربما لا نستطيع توليد **جميع عناوين المصفوفة** لكي نختبرها ← الحل لهذه المشكلة هو حجز طول مصفوفة برقم أولي (prime number) عندها يمكن الوصول إلى نصف عناصر المصفوفة على الأقل وهو عدد معقول.

3- الفحص بإزاحة للمفتاح (Key Offset Probe)

- عند حصول تصادم نحسب العنوان البديل كالتالي:

  New address = (old address +offset) mod list_size

  offset = key / list_size (قسمة صحيحة)

  - حيث أن وجود **قيمة المفتاح** في القانون يسبب مسارات تصادم مختلفة لمفاتيح مختلفة.←نوع التجمع secondary clustering .

- مثال: بفرض طول المصفوفة 307 :

  hash(070918) = 070918 mod 307 = 1

  hash(166702) = 166702 mod 307 = 1 → collision

  offset = 166702 / 307 = 543

  New address = (543 + 1) mod 307 = 237

3- الفحص بإزاحة للمفتاح (Key Offset Probe)

- لرؤية تأثير طريقة إزاحة المفتاح على مسار التصادمات، نقوم بتقطيع عدة مفاتيح مختلفة، وكلها مرتبطة بنفس العنوان الأساسي (home address).
  - يوضح الجدول التالي مسارات ثلاثة مفاتيح تتصادم عند العنوان 001 :

Key	Home Address	Key Offset	Probe 1	Probe 2
166702	1	543	237	166
572556	1	1865	024	047
067234	1	219	220	132

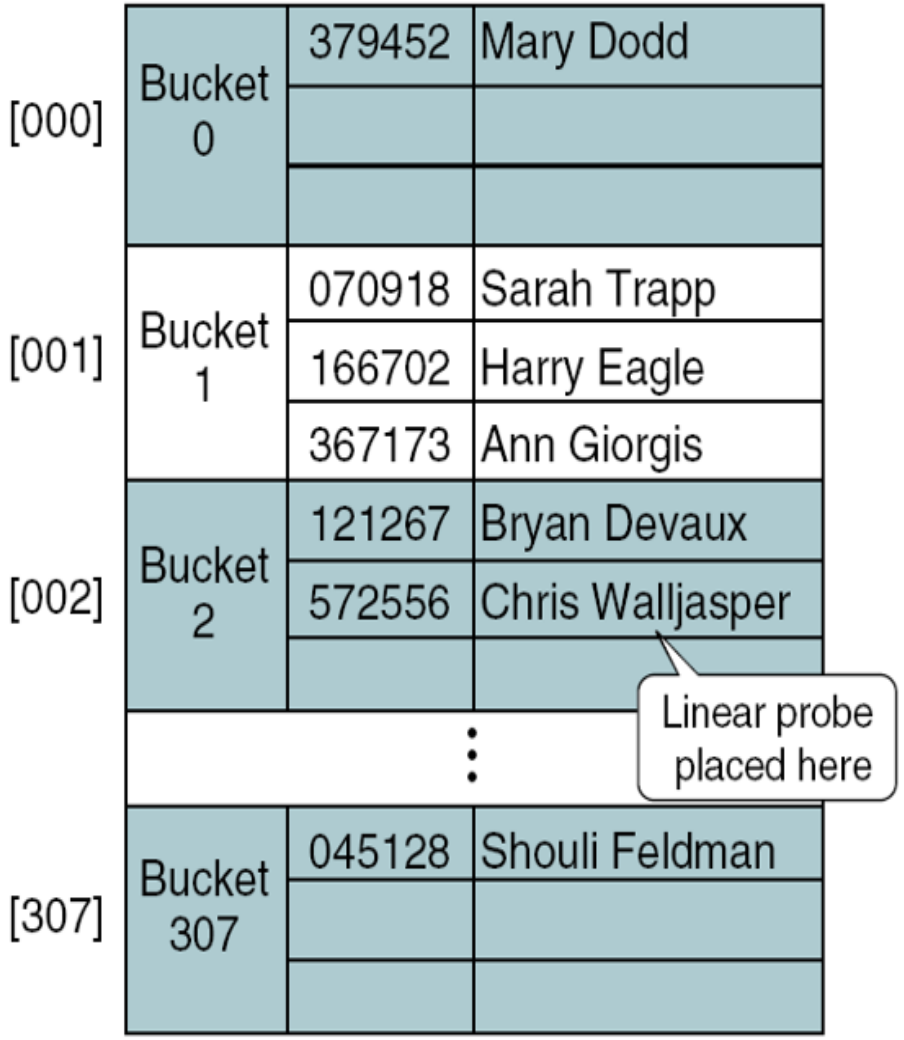
  - نلاحظ أن كل مفتاح يحل تصادمه **بعنوان بديل مختلف** في كل من الفحص الأول والثاني.

التقطيع باستخدام الرزم (Bucket Hashing)

- طريقة أخرى للتعامل مع **مشكلة التصادمات**، حيث
  - يُقسم hash table إلى رزم (buckets).
  - يمكن لكل رزمة أن تستوعب **عدة مفاتيح كلها تحت نفس العنوان** (أي تُجمع عدة مرادفات في رزمة واحدة).
  - هكذا يتم تأجيل حصول تصادم إلى أن تمتلئ الرزمة.

- هناك مشكلتان مع هذا الطريقة.
  - أولاً، يتم **حجز مساحة أكبر** مع احتمال بقاء العديد من الرزم فارغة أو فارغة جزئياً.
  - ثانياً، لا تُحل مشكلة التصادمات تماماً. في حال **امتلاء رزمة** وحصول تصادم نحتاج إلى حساب عنوان لرزمة بديلة.
    - ✓ عند حدوث التصادم، يمكن استخدام أي من طرق حل التصادمات السابقة.

التقطيع باستخدام الرزم (Bucket Hashing)



- **مثال:**
  - نلاحظ أن الرزمة الثانية (Bucket 1) تحتوي على بيانات لثلاث سجلات، وكلها مقطعة إلى نفس العنوان 1 (يعني هي عبارة عن ثلاثة مرادفات).
  - عند حصول تصادم المفتاح 572556 في Bucket 1 يمكن أن نستخدم طريقة الفحص الخطي (linear probe) ونصل على عنوان بديل 2 .

- **ملاحظة:** لتسريع البحث عن مفتاح ضمن الرزمة الواحدة يمكن إدخال المرادفات إلى الرزم بشكل **مرتب تصاعدياً**.

أمثلة محلولة

مثال(1)

- بفرض أنه لدينا جدول تقطيع طوله (7) فارغ تماماً، ولدينا المفاتيح التالية نريد تقطيعها:

18, 12, 25, 47, 16

والمطلوب: ارسم جدول التقطيع الناتج مستخدماً **طريقة باقي القسمة** للتقطيع **وطريقة الفحص الخطي** (Linear probe) لحل التصادمات.

H(18) = 18 % 7 = 4
H(12) = 12 % 7 = 5
H(25) = 25 % 7 = 4 → collision
NewAddress= oldAddress + 1
NewAddress= 4 + 1 = 5 → collision
NewAddress= 5 + 1 = 6

H(47) = 47 % 7 = 5 → collision
NewAddress= 5 + 1 = 6 → collision
NewAddress= 6 + 1 = 7 → out of range
NewAddress= 0

H(16) = 16 % 7 = 2

0	47
1	
2	16
3	
4	18
5	12
6	25

مثال(2)

- ليكن لدينا جدول تقطيع طوله (1000) ويحوي حالياً (4) مفاتيح فقط، حيث تم الاعتماد على طريقة اختيار الأرقام للتقطيع (الخانة الثانية، الثالثة والخامسة) وطريقة الاختبار بإزاحة المفتاح لحل التصادمات.

address	0	...	103	...	466	...	735	...	946	...
Key			41073296		54686930		17345082		29426813	

1. قم بتقطيع المفاتيح التالية لتضاف إلى الجدول السابق: 24606334, 31053809, 84616004

24606334 : H(24606334) = 466 → collision
NewAddress= ((key / listsize) + OldAddress) % listsize
NewAddress= ((24606334 / 1000) + 466) % 1000 = 072

31053809 : H(31053809) = 103 → collision
NewAddress= ((31053809 / 1000) + 103) % 1000 = 156

84616004 : H(84616004) = 466 → collision
NewAddress= ((84616004 / 1000) + 466) % 1000 = 082

2. ما هو نوع التجمع الناتج في الجدول السابق؟ الجواب: secondary clustering

3. احسب قيمة معامل التحميل للجدول الناتج (بعد تنفيذ الطلب الأول): $\alpha = (7/1000)*100 = 0.7\ \%$

مثال(3)

- بفرض أنه نريد تخزين 300 مفتاح في جدول تقطيع مؤلف من رزم (حجم كل رزمة يساوي 2) وذلك مستخدماً كلٍ من طريقة الطوي Fold–Shift للتقطيع والفحص بإزاحة المفتاح لحل التصادمات. والمطلوب:

1- ما هو أنسب طول للجدول يمكن حجزه في هذه الحالة ؟

الجواب: list_size = 1000 لأن عدد المفاتيح بالمئات ونريد استخدام طريقة الطوي [999 .. 0] .

2- قم بتقطيع المفاتيح التالية:

(1) hash(306041220) = 306 + 041 + 220 = 567
(2) hash(110320137) = 110 + 320 + 137 = 567
(3) hash(254102212) = 254 + 102 + 212 = 568

(4) hash(804442321) = 804 + 442 + 321 = 1567 = 567 → collision
NewAddress = ((key / listsize) + OldAddress) % listsize
NewAddress = ((804442321 / 1000) + 567) % 1000 = 009

(5) hash(537020010) = 537 + 020 + 010 = 567 → collision
NewAddress = ((537020010 / 1000) + 567) % 1000 = 587

3- ما نوع التجمع الناتج في الجدول السابق؟ الجواب: secondary clustering.

4- احسب قيمة معامل التحميل للجدول الناتج (بعد تنفيذ الطلب الثاني): $\alpha = (5/1000)*100 = 0.5\ \%$

مثال(4)

- بفرض أنه لدينا جدول تقطيع طوله 100 (أي أن مجال العناوين [99 .. 0]) , والمطلوب:

1- قم بإضافة المفاتيح التالية إلى الجدول مستخدماً كلٍ من طريقة الدوران والطوي للتقطيع والفحص التربيعي لحل التصادم.

Rotation(987654321) = 198765432
hash(198765432) = 19 + 87 + 65 + 43 + 02 = 216 → address = 16

Rotation(100345012) = 210034501
hash(210034501) = 21 + 00 + 34 + 50 + 01 = 106 → address = 06

Rotation(41021127) = 74102112
hash(74102112) = 74 + 10 + 21 + 12 = 117 → address = 17

Rotation(61030204) = 46103020
hash(46103020) = 46 + 10 + 30 + 20 = 106 → address = 06 → collision
NewAddress= OldAddress + 1^2
NewAddress= 6 + 1 = 07

Rotation(30220415) = 53022041
hash(53022041) = 53 + 02 + 20 + 41 = 116 → address = 16 → collision_1
NewAddress= 16 + 1^2 = 17 → collision_2
NewAddress= 17 + 2^2 = 21

2- ما نوع التجمع الناتج في الجدول السابق ولماذا؟ الجواب: secondary clustering بسبب الفحص التربيعي.

3- احسب قيمة معامل التحميل للجدول الناتج (بعد تنفيذ الطلب الأول): $\alpha = (5/100)*100 = 5\ \%$